- TD 3 -
Versionning & build

L'objectif est de réaliser une première version du projet et de maitriser les outils nécessaires pour le travail collaboratif. Il faudra mettre en place un dépôt distant qui sera utiliser par la suite pour travailler en équipe sur les versions suivantes du projet. On développera également les scripts de build et voir comment utiliser l'IDE Eclipse dans ce contexte.

1.Création et utilisation d'un dépôt `git`

Une solution possible pour le dépôt central est d'utiliser le serveur GitHub qui propose un service d'hébergement des dépôts git ainsi qu'une interface web d'administration de votre compte et des dépôts.

1.1.Création d'un compte

Dans un premier temps il faut s'inscrire et éventuellement déposer sa clé publique. Si vous disposez déjà d'un couple de clés `ssh` publique/privée vous pouvez les utiliser; sinon il faudra les créer (avec la commande `ssh-keygen -t rsa`) et déposer sa clé publique (disponible dans le fichier `id_rsa.pub` dans le répertoire `.ssh`). Si vous souhaitez accéder à vos dépôts Bitbucket à partir de machines différentes, vous devrez refaire cette opération pour chaque machine.

1.2.Création d'un dépôt

Vous pouvez créer un dépôt (repository) pour chacun de vos projets - chaque repository peut être partagé avec d'autres utilisateurs.

1.3.Clonage d'un dépôt

Une fois le repository créé vous pouvez créer une copie locale du dépôt distant en utilisant la commande

```
git clone [url]
```

La commande complète peut être obtenue à partir du bouton `Clone` ; mode `SSH` si vous avez déposé la clé publique ou `HTTPS` sinon (dans ce dernier cas il faudra saisir le mot de passe pour chaque commande `git`).

Le répertoire correspondant ainsi que la structure git sont créés en local mais le dépôt est vide puisqu'aucun fichier n'y a encore été déposé.

1.4.Ajouts et modifications dans le dépôt

On peut maintenant ajouter des fichiers et les modifier. On peut à tout moment voir l'état du dépôt en utilisant la commande

```
git status
```

On va ajouter un fichier README qui sera modifié par la suite par les différents participants au projet (les utilisateurs avec qui on a partagé le repository). Dans un premier temps on va crée le fichier README qui contiendra une seule ligne avec votre nom et prénom (e.g. James Bond).

Par défaut, un fichier n'est pas suivi (tracked) - il faut demander explicitement qu'il participe au(x) prochain(s) commit(s) :

`git add README`

On peut maintenant répercuter les modifications des fichiers suivis dans le dépôt local avec la commande

`git commit -m « premiere version »`

On peut faire des modifications successives et les enregistrer dans le dépôt. On peut également consulter les modifications et leur enregistrement (`git log/status/show`).

Le fichier est disponible seulement dans le dépôt local et pas encore dans le dépôt distant. Pour répercuter les modifications dans le dépôt local il faut faire

`git push`

1.5.Travail collaboratif

On peut faire une copie (clone) du même dépôt distant sur une autre machine (celle du binôme) et ensuite faire des modifications sur le fichier README (par exemple, ajouter en deuxième ligne le nom du deuxième participant au projet) et les répercuter dans le dépôt local et à terme dans le dépôt distant. Il faudra faire des modifications en parallèle sur les deux copies et répercuter ces modifications. Il faudra également résoudre les éventuels conflits (par exemple, si chaque participant ajoute une troisième ligne avec un nom différent).

1.6.Ajouter des étiquettes

Dès que le fichier README est dans un état stable et définitif on peut ajouter une étiquette pour marque cette première version stable du projet:

`git tag -a v0.0 -m « meta-info »`

et répercuter l'étiquetage localement avec un `commit` et à distance avec un `push`.

1.7.Ajouter des étiquettes

Vous pouvez revenir à une version précédente en utilisant soit l'identifiant du commit

`git checkout <``id-commit`**`>`**

soit l'identifiant du tag

`git checkout tags/<``id-tag`**`>`**

1.8.Utilisation d'un IDE

L'application peut être également développé dans un IDE (e.g. Eclipse). Pour cela il faut:

- créer un dépôt github (voir plus haut), appelé `ISN` (ajouter éventuellement un README)
- créer un projet Java en Eclipse, appelé `Play`,
- et ajouter une classe (`Hello.java`) dans un package (`simple`)
- dans le menu correspondant pour le projet `Play` faire
  - `Team/Share Project` et
  - créer un dépôt avec le bouton `Create` (appelé `Play`)
- dans le menu correspondant pour le projet `Play` faire
  - `Team/Commit` et dans la fenêtre `Git Staging` ajouter `Hello.java` au `Staged Changes`,
  - appuyer sur `Commit`
- dans le menu correspondant pour le projet `Play` faire
  - `Team/Pull` et dans la fenêtre spécifier les attributs du dépôt GitHub distant, c.a.d.
  - URI obtenu du bouton `Clone or download` de GitHub (mode HTTPS)
  - les identifiants

2.Projet collaboratif

On utilise maintenant le dépôt créé pour gérer les sources du projet à réaliser en équipe. Le nom du dépôt doit avoir la forme **`ACL20xx_nomEquipe`** et donner le droit de lecture à votre responsable de groupe.

2.1.Ajouter les sources

Dans un premier temps il faudra réaliser une première version très simple avec un minimum de fonctionnalités. Il faudra néanmoins bien réfléchir à l'organisation (en packages) des sources pour minimiser les modifications ultérieures. Distribuez ensuite le travail (le développement des classes correspondantes) entre les membres de l'équipe.

> *Rappel*: répercuter (dans le dépôt distant) les modifications d'une classe seulement si le projet compile et s'exécute toujours correctement.

2.2.Build automatique

L'application doit disposer d'un script `ant` (ou `maven`) permettant l'automatisation de la compilation, du test, du packaging (en un `jar`) et de l'exécution. Le projet peut avoir l'organisation proposée en cours ou une organisation similaire de la forme:

```
build.xml
src
 |--- fr/ul/acl/model
                  |--- Pacman.java
                  |--- ...
 |--- fr/ul/acl/start
                  |--- Main.java
                  |--- ...
 |--- fr/ul/acl/tests
                  |--- PacmanTest.java
                  |--- ...
```